In the Specification

Please replace paragraph [0018] in its entirety with the following:

[0018] Turning now to Figure 1 a block diagram of an application monitor system 10 for accessing the value of internal application variables is depicted. A first application 12 is programmed or coded to create a first shared memory 14 and to write values into variables var1 16a, var2 16b, and varN 16c contained within the first shared memory 14. In addition, the first application 12 is coded to perform the task allocated to the first application 12. In the present embodiment, the first application 12 runs in a normal, real-time operational mode, as opposed to running under a debugger or other tool employed to encapsulate the first application 12. While the first application 12 is in real-time operation, whether during testing or other simulated conditions, or otherwise running, a monitor module 18 is operable to read the values of the variables 16 from the first shared memory 14 and is in communication with a client 20. A second application 22 is programmed or coded to create a second shared memory 24 and to write values into variables varA 26a, varB 26b, and varX 26c contained within the second shared memory 24. The monitor module 18 is operable to read the values of the variables 26 ~~variables 24~~ from the second shared memory 24.

Please replace paragraph [0022] in its entirety with the following:

[0022] The first application 12 is programmed so when it runs it first creates the first shared memory 14 and then writes some variables of interest out to variables 16 in the first shared memory 14 whenever the application changes the value of those variables 16. The action of the first application 12 writing these values to the variables 16 in the first shared memory 14 does not interfere with the normal operation of the first application 12. These variables 16 may include an

identifier, such as a variable name, and may have a value associated with the identifier. Both the identifier and value of each variable 16 are made available for display on the UI 28 through the application monitor system 10 through the monitor module 18 reading these values and identifier of each variable 16 from the first shared memory 14, the monitor module 18 sending these values and variable names to the client 20, and the client 20 sending these values and variable names to the user interface 28. If the first application 12 does not write an internal variable to the first shared memory 14, then this internal variable is not accessible to the monitor module 18 and hence not accessible to the client 20. For example, if EmployeeSocialSecurityNumber is an internal variable of the first application 12 and the first application 12 does not write the value of EmployeeSocialSecurityNumber to the first shared memory 14, then the monitor module 18 may not be able to access the value of EmployeeSocialSecurityNumber.

Please replace paragraph [0026] in its entirety with the following:

[0026] In some embodiments the monitor module 18 and the client 20 may be programmed in the Common Business Oriented Language (COBOL) programming language. Unfortunately COBOL may not provide support for sockets or for shared memory. In this case some embodiments may employ a technical layer which provides a means for COBOL programs to avail themselves of certain features, including sockets and shared memory, which otherwise would not be accessible to these COBOL programs. The technical layer in this example provides a plurality of routines including a shared memory routine and a sockets routine. The shared memory routine provides the COBOL program with the functionality to enable memory sharing on the computer 30 between applications, for example between the applications 12, ~~20,~~ 22 and the monitor module 18. The socket routine enables the COBOL program to communicate information between applications on

the same computer and between applications on different computers, for example between the monitor module 18 and the client 20.

Please replace paragraph [0032] in its entirety with the following:

[0032] The second COBOL program 62, using the identifier such as the name, requests the address 56 of the memory 58 wherein the shared memory is located. The shared memory routine 50 returns the address 56 in a similar manner based on the use of the same identifier or name. The address 56 of the shared memory is mapped back to the linkage section 66b of the second COBOL program 62 thereby creating a shared block of memory useable by both the first and second COBOL programs 60 and 62 ~~50 and 52~~.

Please replace paragraph [0045] in its entirety with the following:

[0045] The client 20 may also request the values of variables [[116]] 110, such as ~~var3 116a~~ var3 110c, from the second application 22. The monitor module 18 is also operable to attach to a second memory area 114 of the second application 22. The second application 22 writes values into the second memory area 114 during the normal course of executing. The monitor module 18 is operable to read the values of the variables 110 ~~116~~ from the second memory area 114 and to send these values to the client 20. Note that while two applications are depicted in Figure 4, any number of applications executing on the computer system 30 may be concurrently monitored by the monitor module 18. Also note that while the client 20 is depicted in communication with only one monitor module 18 in Figure 4, the client 20 may communicate concurrently with multiple monitors 18.

Please replace paragraph [0046] in its entirety with the following:

[0046] A user interface 28 is in communication with the client 20. The user interface 28 is operable to select the applications ~~20,~~12, 22 and to select the variables 110 ~~and 114~~ and to communicate these selections to the client 20. The user interface 28 is further operable to receive the values of variables 110 ~~and 114~~ from the client 20 and to display these values.

Please replace paragraph [0047] in its entirety with the following:

[0047] The applications ~~20,~~12, 22 and the monitor module 18 are computer programs or applications which execute on a general purpose computer system 30. The client 20 and the user interface 28 are computer programs or applications which execute on a general purpose computer system 32. General purpose computer systems will be discussed in more detail hereinafter.

Please replace paragraph [0049] in its entirety with the following:

[0049] The applications 12, 22 ~~20~~ write into the memory areas 112, 114 while executing. Note that in this embodiment it is not necessary for the applications ~~20,~~12, 22 to mirror the values of internal variables by writing duplicate values in shared memory regions 14, 24~~18, 26~~. In this embodiment the applications ~~20,~~12, 22 do not need to be changed to enable the operation of the application monitor system 10. This aspect of this embodiment is useful if different user communities employ the monitor at different stages of the software lifecycle. For example, the variables which the original software developer may be interested in during the test phase may not be those that a business analyst is interested in when the software is deployed and in service. Because all the application 12, 22 variables are accessible in the memory areas 112 and 114, the needs of various user communities are readily satisfied.

Please replace paragraph [0051] in its entirety with the following:

[0051] A first address map 116 is stored in secondary storage 118. This secondary storage may be closely associated with computer system 30, with computer system 32, or with some other remote computer system. The first address map 116 contains a listing of all variables names associated with the first application 12 and their address offset relative to the first instruction in the first application 12 executable file. The address offset is used to generate an address to read the value of the variable from the first memory area 112. Similarly, a second address map 120 is stored in secondary storage 118 which contains a listing of all variable names associated with the second application 22 and their address offset relative to the beginning of the second application's 22 address space. The address maps 116 and 120 may be generated when compiling and linking the applications.

Please replace paragraph [0054] in its entirety with the following:

[0054] In some embodiments the client 20 may send the variables 110~~12, 14~~, including the variable name and variable value, to the user interface 28 in extensible markup language (XML) format. XML supports flexible formatting of the variable names and variable values for transmission to the user interface 28.

Please replace paragraph [0055] in its entirety with the following:

[0055] The client and monitor may communicate using a socket connection. When sockets are employed, the monitor module 18 acts as a socket server and the client 20 acts as a socket client. As discussed under the shared memory embodiment of the application monitor system 10, when

COBOL is employed as the programming language for encoding the applications [[20]]12 and 22 a technical layer may be employed to provide convenient access to operating system socket services.

Please replace paragraph [0056] in its entirety with the following:

[0056] The embodiment of the application monitor system 10 where on the monitor module 18 attaches to the first memory area 112 of the first application 12 may be implemented in any programming language and is not limited to being implemented in COBOL. The applications 12, 22 monitored by the application monitor system 10 may be implemented in any programming language and need not be implemented in COBOL.

Please replace paragraph [0057] in its entirety with the following:

[0057] The application monitor system 10 described above may be implemented on any general-purpose computer with sufficient processing power, memory resources, and network throughput capability to handle the necessary workload placed upon it. Figure [[4]] 5 illustrates a typical, general-purpose computer system suitable for implementing one or more embodiments disclosed herein. The computer system 380 includes a processor 382 (which may be referred to as a central processor unit or CPU) that is in communication with memory devices including secondary storage 384, read only memory (ROM) 386, random access memory (RAM) 388, input/output (I/O) 390 devices, and network connectivity devices 392. The processor may be implemented as one or more CPU chips.